2100030336

M. Gayathri Ankitha

.NET PROGRAMMING

LAB-4

IN-LAB:

Task1: To create classes Employee, SalesPerson, Manager and Company with predefined functionality.

Low level requires:

1. To create basic class **Employee** and declare following content:

- Three closed fields – text field **name** (employee last name), money fields – **salary** and **bonus**
- Public property **Name** for reading employee's last name
- Public property **Salary** for reading and recording salary field
- Constructor with parameters string **name** and money **salary** (last name and salary are set)
- Virtual method **SetBonus** that sets bonuses to salary, amount of which is delegated/conveyed as bonus
- Method ToPay that returns the value of summarized salary and bonus.

2. To create class **SalesPerson** as class **Employee** inheritor and declare within it:

- Closed integer field **percent** (percent of sales targets plan performance/execution)
- Constructor with parameters: **name** – employee last name, **salary**, **percent** – percent of plan performance, first two of which are passed to basic class constructor
- Redefine virtual method of parent class **SetBonus** in the following way: if the sales person completed the plan more than 100%, so his bonus is doubled (is multiplied by 2), and if more than 200% - bonus is tripled (is multiplied by 3)

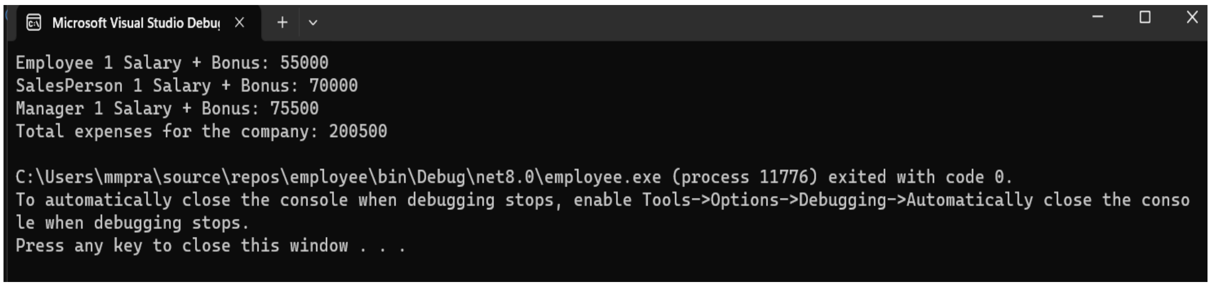
3. To create class **Manager** as **Employee** class inheritor, and declare with it:

- Closed integer field **quantity** (number of clients, who were served by the manager during a month)

- Constructor with parameters string **name** – employee last name, **salary** and integer **clientAmount** – number of served clients, first two of which are passed to basic class constructor.
- Redefine virtual method of parent class **SetBonus** in the following way: if the manager served over 100 clients, his bonus is increased by 500, and if more than 150 clients – by 1000.

```
using System;
public class Employee
{
    private string name;
    private decimal salary;
    private decimal bonus;
    public string Name { get { return name; } }
    public decimal Salary { get { return salary; } }
    public Employee(string name, decimal salary)
    {
        this.name = name;
        this.salary = salary;
    }
    public virtual void SetBonus(decimal bonus)
    {
        this.bonus = bonus;
    }
    public decimal ToPay()
    {
        return salary + bonus;
    }
}
public class SalesPerson : Employee
{
    private int percent;
    public SalesPerson(string name, decimal salary, int percent) : base(name, salary)
    {
        this.percent = percent;
    }
    public override void SetBonus(decimal bonus)
    {
        if (percent > 200)
        {
            base.SetBonus(bonus * 3);
        }
        else if (percent > 100)
        {
            base.SetBonus(bonus * 2);
        }
        else
        {
            base.SetBonus(bonus);
        }
    }
}
public class Manager : Employee
{
```

```
•     private int quantity;
•     public Manager(string name, decimal salary, int clientAmount) :
  base(name, salary)
•     {
•         this.quantity = clientAmount;
•     }
•     public override void SetBonus(decimal bonus)
•     {
•         if (quantity > 150)
•         {
•             base.SetBonus(bonus + 1000);
•         }
•         else if (quantity > 100)
•         {
•             base.SetBonus(bonus + 500);
•         }
•         else
•         {
•             base.SetBonus(bonus);
•         }
•     }
• }
• public class Company
• {
•     public decimal CalculateTotalExpenses(Employee[] employees)
•     {
•         decimal totalExpenses = 0;
•         foreach (var emp in employees)
•         {
•             totalExpenses += emp.ToPay();
•         }
•         return totalExpenses;
•     }
• }
• class Program
• {
•     static void Main(string[] args)
•     {
•         Employee emp1 = new Employee("John", 50000);
•         SalesPerson sp1 = new SalesPerson("Alice", 60000, 150);
•         Manager manager1 = new Manager("Bob", 70000, 120);
•         emp1.SetBonus(5000);
•         sp1.SetBonus(5000);
•         manager1.SetBonus(5000);
•         Console.WriteLine("Employee 1 Salary + Bonus: " + emp1.ToPay());
•         Console.WriteLine("SalesPerson 1 Salary + Bonus: " + sp1.ToPay());
•         Console.WriteLine("Manager 1 Salary + Bonus: " +
  manager1.ToPay());
•         Company company = new Company();
•         Employee[] employees = { emp1, sp1, manager1 };
•         decimal totalExpenses = company.CalculateTotalExpenses(employees);
•         Console.WriteLine("Total expenses for the company: " +
  totalExpenses);
•     }
• }
```



```
Employee 1 Salary + Bonus: 55000
SalesPerson 1 Salary + Bonus: 70000
Manager 1 Salary + Bonus: 75500
Total expenses for the company: 200500

C:\Users\mmpra\source\repos\employee\bin\Debug\net8.0\employee.exe (process 11776) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

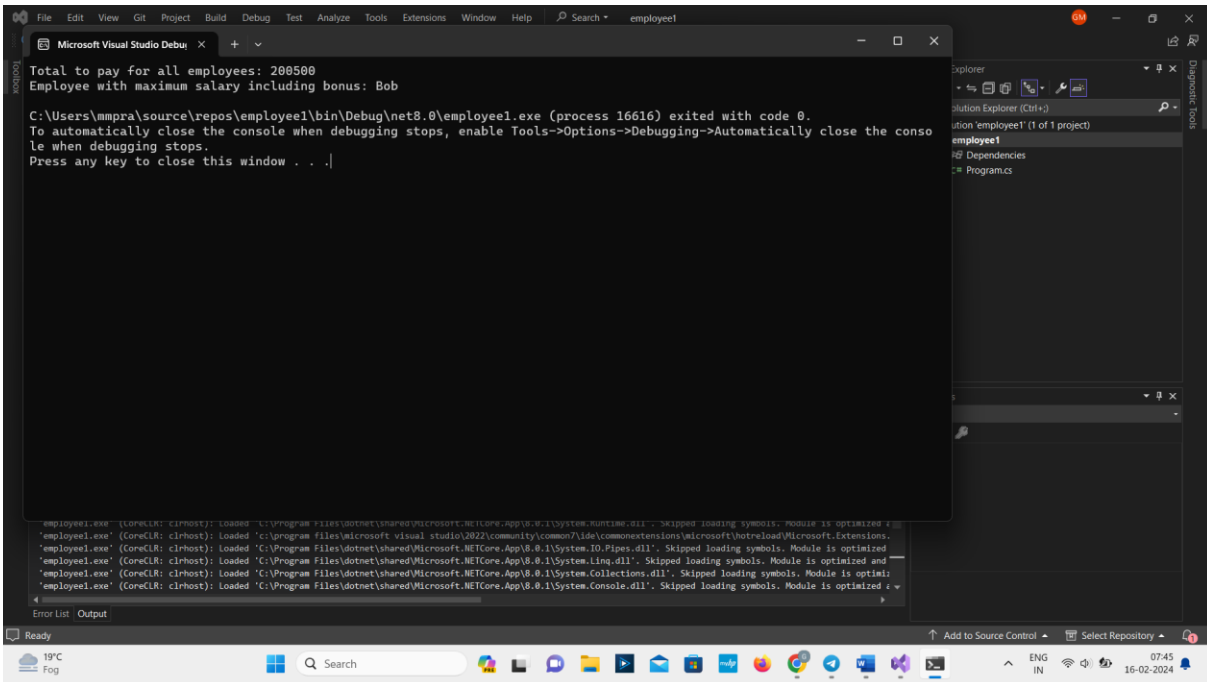
TASK 2: Advanced level requires:

1. To fully complete Low level tasks.
2. Create class Company and declare within it:

- Closed field **employees** (staff) – an array of Employee type.
- Constructor that receives employee array of **Employee** type with arbitrary length
- Method **GiveEverybodyBonus** with money parameter **companyBonus** that sets the amount of basic bonus for each employee.
- Method **TotalToPay** that returns total amount of salary of all employees including awarded bonus
- Method **NameMaxSalary** that returns employee last name, who received maximum salary including bonus.

```
using System;
public class Employee
{
    private string name;
    private decimal salary;
    private decimal bonus;
    public string Name { get { return name; } }
    public decimal Salary { get { return salary; } }
    public Employee(string name, decimal salary)
    {
        this.name = name;
        this.salary = salary;
    }
    public virtual void SetBonus(decimal bonus)
    {
        this.bonus = bonus;
    }
    public decimal ToPay()
    {
        return salary + bonus;
    }
}
public class SalesPerson : Employee
{
    private int percent;
    public SalesPerson(string name, decimal salary, int percent) : base(name,
salary)
    {
        this.percent = percent;
    }
    public override void SetBonus(decimal bonus)
    {
        if (percent > 200)
        {
            base.SetBonus(bonus * 3);
        }
        else if (percent > 100)
        {
            base.SetBonus(bonus * 2);
        }
        else
        {
            base.SetBonus(bonus);
        }
    }
}
public class Manager : Employee
{
    private int quantity;
    public Manager(string name, decimal salary, int clientAmount) : base(name,
salary)
    {
        this.quantity = clientAmount;
    }
    public override void SetBonus(decimal bonus)
    {
        if (quantity > 150)
        {
            base.SetBonus(bonus + 1000);
        }
        else if (quantity > 100)
        {
            base.SetBonus(bonus + 500);
        }
        else
        {
            base.SetBonus(bonus);
        }
    }
}
public class Company
{
    private Employee[] employees;
    public Company(Employee[] employees)
    {
        this.employees = employees;
    }
    public void GiveEverybodyBonus(decimal companyBonus)
    {
        foreach (var emp in employees)
        {
            emp.SetBonus(companyBonus);
        }
    }
    public decimal TotalToPay()
    {
        decimal totalToPay = 0;
        foreach (var emp in employees)
        {
            totalToPay += emp.ToPay();
        }
        return totalToPay;
```

```csharp
        }
        public string NameMaxSalary()
        {
            decimal maxSalary = 0;
            string maxSalaryEmployee = "";
            foreach (var emp in employees)
            {
                decimal empSalary = emp.ToPay();
                if (empSalary > maxSalary)
                {
                    maxSalary = empSalary;
                    maxSalaryEmployee = emp.Name;
                }
            }
            return maxSalaryEmployee;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Employee emp1 = new Employee("John", 50000);
            SalesPerson sp1 = new SalesPerson("Alice", 60000, 150);
            Manager manager1 = new Manager("Bob", 70000, 120);
            Employee[] employees = { emp1, sp1, manager1 };
            Company company = new Company(employees);
            company.GiveEverybodyBonus(5000);
            Console.WriteLine("Total to pay for all employees: " +
    company.TotalToPay());
            Console.WriteLine("Employee with maximum salary including bonus: " +
    company.NameMaxSalary());
        }
      }
```



```
Total to pay for all employees: 200500
Employee with maximum salary including bonus: Bob

C:\Users\mmpra\source\repos\employee1\bin\Debug\net8.0\employee1.exe (process 16616) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

```
'employee1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.1\System.Runtime.dll'. Skipped loading symbols. Module is optimized a
'employee1.exe' (CoreCLR: clrhost): Loaded 'c:\program files\microsoft visual studio\2022\community\common7\ide\commonextensions\microsoft\hotreload\Microsoft.Extensions.
'employee1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.1\System.IO.Pipes.dll'. Skipped loading symbols. Module is optimized
'employee1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.1\System.Linq.dll'. Skipped loading symbols. Module is optimized and
'employee1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.1\System.Collections.dll'. Skipped loading symbols. Module is optimiz
'employee1.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.1\System.Console.dll'. Skipped loading symbols. Module is optimized a
```